**Data Communication and Networks**

**Q-1 Role of router in network security**

* Everyone uses an Internetwork router to connect to the Internet. A router's first job is to route, transparently and seamlessly directing packets from one network to another.
* But a router can do much more. First of all, if you know how to describe "bad" behavior, a router can look for it in Internetwork traffic. For example, if you can associate certain IP addresses with the network interfaces of a router, the router can tell you if an outside computer is pretending to be inside your network--a classic IP spoofing attack.
* Routers can also be configured to address source-routed address requests in packets. These are packets that basically say, "You see where it has my IP address here in this field? Well, when you send packets back to me, don't check your routing table or anyone else's to send me the packet. Instead, send it to this other address here."
* Another [security feature of routers](https://searchsecurity.techtarget.com/answer/Can-the-PORTAL-travel-router-improve-traffic-security) is the ability to filter. Filtering applies policy to packets, declaring what is permitted and denied by using rules that specify...
* Network interface: Which network did this packet come from?
* Source: What IP address did it come from?
* Destination: Where does it want to go?
* Packet type
* Protocol: What language to talk--for example, HTTP for Web traffic or SMTP for e-mail.
* Port to use: matches the packet with a particular service running on a computer--for example, e-mail is usually on port 25, Web runs over port 80.
* Picture a simple Internet gateway to a private network. The connection to the Internet is a border router whose internal interface is connected to our service network--the DMZ. Connected to that same network are a Web server and an enterprise firewall, which in turn is connected to the private network.
* First, the router should be configured to block stupid hacker tricks, such as IP spoofing. Though the firewall could certainly handle these attacks, why bother it with this "kid stuff?" Blocking such attacks at the router means the Web server doesn't have to worry about whether the packet address is spoofed.
* Also, the security policy should list what network services are allowed between our networks and the Internet. The router can be used to allow required services and deny everything else.
* Next, we can tune the router to the needs of the DMZ. What traffic should be going to the Web server in the DMZ? Answer: nothing but Web traffic. So, we configure the router to allow HTTP and SSL traffic to ports 80 and 443 from the Internet to the Web server, and nothing else.
* How about packets originating at the Web server hitting the router on the way to the Internet? There shouldn't be any, right? Only responses to outside requests should be permitted. So, we tell the router to permit only HTTP and SSL from the router to the Internet, and only in response to outside requests.
* The enterprise firewall is also on the DMZ. The router filters should be configured to deny anything that the enterprise firewall denies. So, even though our enterprise firewall can detect a port scan, we tell the router to look for this common practice, which can be a prelude to an attack. That way, if the firewall gets port-scanned, we know something is very, very wrong. Port-scanning the router may be commonplace--for many organizations, it's a daily occurrence. But if someone is able to port-scan the firewall, we know he has effectively circumvented the router's controls, which should sound a "this should never happen" alarm.
* Yes, "belt and suspenders" is an old idea. But in an era when more servers and applications are being opened up to the Internet, we have to rely on complementary security controls to reduce our risk while maintaining usability.

**Q-2 Explain sliding window protocol with example.**

* Frames have sequence number 0 to maximum 2n - 1 (n bit field).
At any moment, the sender maintains a list of sequence numbers it is permitted to send - these fall within the **sending window**. These are frames sent-but-no-ack and frames not-yet-sent.
When new packet from Network layer comes in to send, it is given highest no, and upper edge of window advanced by 1.
When ack comes in, lower edge of window advanced by 1.
* Receiver has **receiving window** - the frames it is permitted to accept.


Sliding window size 1. Sequence nos. 0 to 7.
(a) At start. Receiver waits for 0.
(b) Sender sends 0.
(c) Receiver receives 0. Waits for 1.
(d) Sender got ack for 0. Hasn't got 1 from its Network layer yet.

* More complex Data Link layer, as more freedom about the order in which it sends and receives frames.
Sender may have n unacknowledged frames at any time (window size n).
Needs n buffers to hold them all for possible re-transmit.
If window grows to its maximum size, DA must shut off NA.
This is all hidden from NB - still receives packets in exact same order.
1. **Sender window might grow** as it receives more frames to send and still has ones un-ack'ed.
Starts with nothing to send, then NA gives it frames to send.
Later, window may shrink as frames are ack-ed and NA has no more.
2. **Receiver window constant size**.
Receiver window size 1 means will only accept them in order.
Size n means will receive out of order (e.g. receive later ones after earlier frame is lost)
and then must *buffer* them before sending to NB (must send to NB in order).

e.g. DB has buffers to receive frames 0..7
Receives 1..7 in varying orders. Still waiting on 0. Can't send frames to NB yet.
0 got lost and was re-sent. Eventually gets 0.
Can now send all of 0..7 to NB
and re-use these slots.

e.g. consider frames numbered 0..7 but DB only has 2 buffers
Currently the sliding window is over 4,5
If get 4 can send it to NB and move window to 5,6
If get 5 have to wait for 4, then send both, and advance window to 6,7

* **Stop-and-wait ARQ**



The following transition may occur in Stop-and-Wait ARQ:

* + The sender maintains a timeout counter.
	+ When a frame is sent, the sender starts the timeout counter.
	+ If acknowledgement of frame comes in time, the sender transmits the next frame in queue.
	+ If acknowledgement does not come in time, the sender assumes that either the frame or its acknowledgement is lost in transit. Sender retransmits the frame and starts the timeout counter.
	+ If a negative acknowledgement is received, the sender retransmits the frame.
* **Go-Back-N ARQ**

Stop and wait ARQ mechanism does not utilize the resources at their best.When the acknowledgement is received, the sender sits idle and does nothing. In Go-Back-N ARQ method, both sender and receiver maintain a window.



* + The sending-window size enables the sender to send multiple frames without receiving the acknowledgement of the previous ones. The receiving-window enables the receiver to receive multiple frames and acknowledge them. The receiver keeps track of incoming frame’s sequence number.
	+ When the sender sends all the frames in window, it checks up to what sequence number it has received positive acknowledgement. If all frames are positively acknowledged, the sender sends next set of frames. If sender finds that it has received NACK or has not receive any ACK for a particular frame, it retransmits all the frames after which it does not receive any positive ACK.
* **Selective Repeat ARQ**
* In Go-back-N ARQ, it is assumed that the receiver does not have any buffer space for its window size and has to process each frame as it comes. This enforces the sender to retransmit all the frames which are not acknowledged.



* In Selective-Repeat ARQ, the receiver while keeping track of sequence numbers, buffers the frames in memory and sends NACK for only frame which is missing or damaged.
* The sender in this case, sends only packet for which NACK is received.

**Q-3 what is socket? Explain fundamental and needs of socket programming.**

* Sockets programming is the fundamental technology behind communications on [TCP/IP](https://www.lifewire.com/transmission-control-protocol-and-internet-protocol-816255) networks. A socket is one endpoint of a two-way link between two programs running on a network. The socket provides a bidirectional communication endpoint for sending and receiving data with another socket. Socket connections normally run between two different computers on a local area network ([LAN](https://www.lifewire.com/local-area-network-816382)) or across the internet, but they can also be used for interprocess communication on a single computer.
* Needs of socket programming
* A socket represents a single connection between exactly two pieces of software (a so-called point-to-point connection). More than two pieces of software can communicate with [client/server](https://www.lifewire.com/introduction-to-client-server-networks-817420) or distributed systems by using multiple sockets. For example, many Web browsers can simultaneously communicate with a single Web server via a group of sockets made on the server.
* Socket-based software usually runs on two separate computers on the network, but sockets can also be used to communicate locally (interprocess) on a single computer. Sockets are bidirectional, meaning that either side of the connection is capable of both sending and receiving data. Sometimes the one application that initiates communication is termed the "client" and the other application the "server," but this terminology leads to confusion in [peer to peer](https://www.lifewire.com/introduction-to-peer-to-peer-networks-817421) networking and should generally be avoided.

**Fundamentals of socket programming**

* Socket is very useful for communication between systems, that's called inter process communication. Sockets can communicate within the systems or across network system.

Socket System consist of client system as well as server system

**Client System**

* Client System acts as a Receiver . In this system Requests to Server, and receive information from Server.​

**Server System**

* Server system acts as a provider of information. It acts as an Administrator, to control over Network Systems ( Clients ). Provide information depending on Client request and responds to the client request.
​

Socket programming, requires you to create two applications
1) One is Server applications.
2) Second one is client applications ( number of clients )

Server application should be capable of handling more than one client if you have a requirement where number of clients will be connecting.

The below table illustrates server and client related socket functions

**Socket Functions**

**Server functions**

* socket
* bind
* listen
* accept
* read
* write
* close

**Client functions**

* socket
* connect
* send
* receive
* close

**Functionalities**

Socket

» To Initialize a Socket​

Bind

» To attach an Interface​

Listen

» Listen for Incoming Connection​

Accept

» Permits an Incoming Connection​

Connect

» Establish a Connection on a Specified socket​

Send

» Send data on Connected Socket​

Only two types of Specifications supported for Windows

**1) Stream Socket**

It provides sequenced, reliable connection oriented byte stream. It uses TCP [ Transmission control protocol ] to transmit data to the connected system in sequenced manner. It also waits for acknowledgement.

**2) Datagram Socket:**

It provides unreliable connection less service. It uses UDP [ User datagram protocol ] transport layer protocol and its not required to connect the network systems.

UDP and TCP use the IP address and the port number to uniquely identify a particular process on a networked host.

**Q-4 How congestion occurs in network? List out step for it’s avoidance.**

Reason behind congestion:

## Too many hosts in broadcast domain

Network overload is what happens when you place too many hosts in a broadcast domain. The term “broadcast domain” is abstract, but the concept applies to network structure.

The broadcast domain is the network. This could be the network in an enterprise, university, or a VLAN. A host is going to be each individual router or switch within the broadcast domain.



* The concept also applies to mobile networks and routers. Mobile networks and routers are the broadcast domain. While computers, tablets, or phones are the hosts.
* Thus, too many hosts in a broadcast domain can create network congestion. The cause is network overload, as too many devices are requesting network access at once.
* A safe number of hosts in a broadcast domain is 200 – 254.

## Broadcast Storms

* A broadcast storm is a situation where there are unexpectedly too many requests on a network. This creates a situation where a network does not have the ability to process all the requests at once.
* A broadcast storm can be a busy day for [eCommerce](https://datapath.io/resources/whitepapers/ecommerce-conversion-optimization-whitepaper/) or Black Friday sales. Also, a video going viral can cause a similar situation.

## Low Bandwidth

* [Bandwidth](https://datapath.io/resources/blog/what-is-bandwidth/) refers to the “size of the pipe” in which Internet data can travel through. If the pipe is not large enough for all the traffic to move through at once, there becomes congestion.
* This occurs during peak TV streaming hours when Netflix is consuming 40% of the Internet. The result is congestion, as many people are trying to consume large file size streaming.



## Adding Retransmitting Hubs

* When building out a network, there needs to be the integration of hubs. Hubs retransmit data over a network.
* In an enterprise network, a hub is what connects the network to the public Internet.
* This connection point offers a prime location for potential congestion. Thus, consider how to integrate the hub within the network.

## Multicasting

* Multicasting is where a network allows many computers to speak to each other simultaneously. This is the opposite of Unicast. Unicast is sending traffic to a specific router associated with a specific server. [Learn what is Anycast](https://datapath.io/products/anycast/).



* Multicasting is designed to end network congestion. The reality is it could be creating it. As with most things, there are unintended consequences.
* In multicasting, two packets transferred at the same time can cause a collision. This collision causes network congestion.

## Outdated Hardware

* Data transmitted through outdated switches, routers, servers, and Internet exchanges can cause bottlenecks.
* If the hardware is not optimal, this creates a bottleneck for the transmission of data. The result is network congestion.

## Bad Configuration Management

* Fat fingers and misconfiguration are two things that can cause network congestion. [NetDevOps](https://datapath.io/resources/blog/achieve-network-automation-netdevops/) has the goal of eliminating this problem.
* Repetitive and one off scripts leaves room for error. The result is a network engineer introducing a bug into the network. These bugs create bottlenecks that cause congestion.
* The other aspect to configuration management is not performing it. This is not maintaining your network. This is like a car. When no maintenance is performed, there is a chance for a break-down.
* With NetDevOps and introducing an element of automation, these instances are reduced.

## Rogue Adapter Broadcasts

* Rogue adapters are any foreign devices on your network. This can be as simple as a neighbor coming onto a residential WiFi connection. Or, as severe as a hacker breaking into an enterprise network.
* What occurs is the rogue adapter finds an entry point, which is usually an error in the network. Then once on the network, they begin to access the Internet. Having an extra device on a network can cause unexpected slowdowns.
* Besides slowing the network, the bigger problem is the security threat. Any foreign device on a network can become malicious in intent.

## Border Gateway Protocol

* [Border Gateway Protocol (BGP)](https://datapath.io/resources/blog/10-things-to-know-about-border-gateway-protocol/) can be causing network congestion. BGP sends all traffic through the shortest logical path. There is no consideration for how much traffic is already going over that path.
* No consideration for current data going can result in transit paths becoming overloaded. This overload will create slower speeds, which is network congestion.

## Artificial Congestion

* An Internet Service Provider (ISP) can determine how fast it sends traffic over its network. The opposite result of this is the ISP can also slow the rate at which data is moving over its network. This is artificial [congestion](https://datapath.io/resources/blog/what-is-network-congestion/). ISP’s do this for many reasons, which they claim as network management.
* What is affecting this is the established peering agreements. Also, a content provider can be sending more traffic than the ISP would like. The larger implications of this is can be an infringement on net neutrality.
* Network congestion can present itself in many forms. By addressing the causes of network congestion, you can begin to improve your network.

 **Congestion avoidance :**

* Congestion can occur when data arrives on a big pipe (a fast LAN) and gets sent out a smaller pipe (a slower WAN). Congestion can also occur when multiple input streams arrive at a router whose output capacity is less than the sum of the inputs. Congestion avoidance is a way to deal with lost packets.
* It is described in [[2](https://tools.ietf.org/html/rfc2001#ref-2)]. The assumption of the algorithm is that packet loss caused by damage is very small (much less than 1%), therefore the loss of a packet signals congestion somewhere in the network between the source and destination. There are two indications of packet loss: a timeout occurring and the receipt of duplicate ACKs.
* Congestion avoidance and slow start are independent algorithms with different objectives. But when congestion occurs TCP must slow down its transmission rate of packets into the network, and then invoke slow start to get things going again. In practice they are implemented together.
* Congestion avoidance and slow start require that two variables be maintained for each connection: a congestion window, cwnd, and a slow start threshold size, ssthresh. The combined algorithm operates as follows:

 1. Initialization for a given connection sets cwnd to one segment and ssthresh to 65535 bytes.

 2. The TCP output routine never sends more than the minimum of cwnd and the receiver's advertised window.

 3. When congestion occurs (indicated by a timeout or the receptionof duplicate ACKs), one-half of the current window size (the minimum of cwnd and the receiver's advertised window, but at least two segments) is saved in ssthresh. Additionally, if the congestion is indicated by a timeout, cwnd is set to one segment

 (i.e., slow start).

 4. When new data is acknowledged by the other end, increase cwnd,

 but the way it increases depends on whether TCP is performing

 slow start or congestion avoidance.

 If cwnd is less than or equal to ssthresh, TCP is in slow start;

 otherwise TCP is performing congestion avoidance. Slow start

 continues until TCP is halfway to where it was when congestion

 occurred (since it recorded half of the window size that caused

 the problem in step 2), and then congestion avoidance takes over.

 Slow start has cwnd begin at one segment, and be incremented by

 one segment every time an ACK is received. As mentioned earlier,

 this opens the window exponentially: send one segment, then two,

 then four, and so on. Congestion avoidance dictates that cwnd be

 incremented by segsize\*segsize/cwnd each time an ACK is received,

 where segsize is the segment size and cwnd is maintained in bytes.

 This is a linear growth of cwnd, compared to slow start's

 exponential growth. The increase in cwnd should be at most one

 segment each round-trip time (regardless how many ACKs are

 received in that RTT), whereas slow start increments cwnd by the

 number of ACKs received in a round-trip time.

**Q-5 Explain original classful addressing scheme in detail.**

There are some classes which are used for addressing a networking nodes.these classis are: A, B, C, D and E. Only classes A, Band C are available for commercial use.

We can find the class of an address when given the address in binary notation or dotted decimal notation.

If the address is given in binary notation, the first few bits can tell us the class of the address.

• If the address is given in dotted decimal notation, the first byte defines the class.

## ****Class A addresses****

1. Class A addresses are designed for large organizations with a large number of hosts or routers.

2. In this the first octet of the address identifies the network and the next three octets are used to identify the host.

3. The first bit of first octet is always 0 and the remaining 7 bits are used to identify the network address.

4. The next three octets *i.e.* 24 bits are used to identify the host.

5. The class support addresses from 0.0.0.0 to 0.255.255.255

6. The first block of network address starts with 1.0.0.0 and the last block of network address starts with 127.0.0.0.

7. As there are 7 bits in network address, 27 = 128 blocks of network address are possible. Out of these two network blocks are reserved. Hence total 126 address blocks are used.

8. Each network blocks can have 224--- 2 hosts *i.e.* 16,777,214 host address. Two addresses are less as one address is reserved for the broadcast address and one address is reserved for the network.

9. A block in class A is too large for almost any organization. This means most of the addresses in class A are wasted and are not used.

## ****Class B address****

1. The class B addresses are designed for medium sized organizations with tens of thousands of attached hosts or routers.

2. In this, the first two octets of the address identify the network and the next two octets identify the host within the network.

3. The first two bits (high order bits) of first octet are always 1,0. Thus the remaining 14 bits identify the network

4. The third and fourth octet *i.e.* 6 bits are used to identify the host.

5. The first network block of this class covers the addresses from 128.0.0.0 to 128.0.255.255 (net id 128.0). The last network block of this class covers addresses from 191.255.255.255 (net id 191.255)

6. The maximum number of network blocks in class B is 214 = 16384.

7 Each network block in class B can have 216--- 2 = 65,534 hosts.

8. A block in class B is also very large and most of the address in class B is also wasted.

## ****Class C address****

1. The class C addresses is designed for small organizations with a small number of attached hosts or routers.

2. In class C, the first three octets of address are used for network and the last octet is used to identify the host.

3. The first three bits of first octet are always set to 1, 1, 0.

4. The remaining 24 - 3 = 21 bits are used for network identification and only 8 bits are used for host.

5. In class C, 221 = 2,097,152 network blocks are possible.

6. Thus, each block in class C address can have 28 - 2 = 254 hosts.

7. The first block of network covers addresses from 192.0.0.0 to 192.0.0.255.

The last block of network covers the addresses form 223.255.255.0 to 223.255.255.255

8. The class C addresses are too less for many organizations as it supports only 254 hosts in a network.

## ****Class D address****

1. Class D addresses are used for multicast groups (multicasting)

2. The concept of division of octets into network id and host id does not apply to class D.

3. The first four bits of first octet in class D are always set to 1,1,1,0.

4. The address range is 224.0.0.0 to 239.255.255.255

## ****Class E address****

1. The Class E address are reserved for future use and are experimental.

2. The concept of network id and hostid does not apply on class E also.

3. The first four bits of first octet are always set to 1,1,1,1.

4. The address range for class E is 240.0.0.0 to 255.255.255.255.



Example :

From the 32 bit IP address we can create dotted decimal notation by converting each byte to a decimal number between 0 and 255. We can also identity the class of an IP address by observing the first few bits of 1st byte of each IP address. The various examples are given in the table below.



We can also find the network address i.e. net id and host id for the IP address (dotted decimal notation) with help of class as shown below.



Explain Ethernet framework structure.



The above Ethernet framework structure is based on IEEE 802.3. Following is the detail of the above structure.

|  |  |  |
| --- | --- | --- |
| PreambleStart frame delimiter (SFD)  | 8 bytes  | Synchronization of the receiversBit sequence that initiates the frame  |
| Destination address (MAC)  | 6 bytes  | Hardware address of the destination network adapter  |
| Source address (MAC)  | 6 bytes  | Hardware address of the source network adapter  |
| Tag  | 4 bytes  | Optional VLAN tag for integration in VLAN networks (IEEE 802.1q)  |
| Type  | 2 bytes  | Ethernet II: labeling of layer 3 protocols  |
| Length  | 2 bytes  | Length information about the record  |
| Destination service access point (DSAP)  | 1 byte  | Individual address of the addressed service access point  |
| Source service access point (SSAP)  | 1 byte  | Source address of the sending device  |
| Control  | 1 byte  | Defines the LLC frame (logical link)  |
| SNAP  | 5 bytes  | Field for the definition of the organizationally unique identifier (OUI) of the manufacturer and the protocol number (like "Type")  |
| Data  | 44-1,500 bytes (limit depending on frame structure)  | The data to be transmitted  |
| Frame check sequence (FCS)  | 4 bytes  | Checksum that computes the entire frame  |
| Inter frame gap (IFS)  | -  | Transmission break of 9.6 μs  |